

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belgaum – 590 018



A project report on

“IoT enabled Multifunctional suit”

submitted in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

INFORMATION SCIENCE & ENGINEERING

by

Adarsh Sharma (1CR16IS006)

Kartik Kashyap (1CR16IS038)

Kartik Kumar (1CR16IS039)

Shashwat Kumar (1CR16IS099)

Under the guidance of

Dr. S. Saravanan

Associate Professor

Dept. of ISE, CMRIT, Bengaluru



CMR INSTITUTE OF TECHNOLOGY
DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

#132, AECS Layout, IT Park Road, Bengaluru-560037

2019-20

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belgaum – 590 018



DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Certificate

This is to certify that the project entitled, “IoT enabled multifunctional suit”, is a bonafide work carried out by **Adarsh Sharma (1CR16IS006)**, **Kartik Kashyap (1CR16IS038)**, **Kartik Kumar (1CR16IS039)**, and **Shashwat Kumar (1CR16IS099)** in partial fulfillment of the award of the degree of Bachelor of Engineering in Information Science & Engineering of Visvesvaraya Technological University, Belgaum, during the year 2019-20. It is certified that all corrections/suggestions indicated during reviews have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the said Degree.

Name & Signature of Guide
(Dr. S. Saravanan)

Name & Signature of HOD
(Dr. M. Farida Begam)

Signature of Principal
(Dr.Sanjay Jain)

External Viva

Name of the Examiners

Signature with date

- 1.
- 2.

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belgaum – 590 018



DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Declaration

We, **Adarsh Sharma (1CR16IS006)**, **Kartik Kashyap (1CR16IS038)**, **Kartik Kumar (1CR16IS039)**, and **Shashwat Kumar (1CR16IS099)**, bonafide students of CMR Institute of Technology, Bangalore, hereby declare that the report entitled “**IoT enabled multifunctional suit**” has been carried out by us under the guidance of **Dr. S. Saravanan, Associate Professor**, CMRIT Bangalore, in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in **Information Science Engineering**, of the Visvesvaraya Technological University, Belgaum during the academic year 2019-2020. The work done in this dissertation report is original and it has not been submitted for any other degree in any university.

Adarsh Sharma (1CR16IS006)

Kartik Kashyap (1CR16IS038)

Kartik Kumar (1CR16IS039)

Shashwat Kumar (1CR16IS099)

ABSTRACT

This project basically deals with design of an IoT enabled multifunctional suit. One of the requirements of IoT that the devices should be small and low-power. Processors and their evolution of the last years, have become a key piece for IoT devices. Intel Edison and Intel Curie are the most recognized technologies in the market and provide all requirements that suit to IoT.

This application is based on this kind of devices to connect the human body to the cloud. The technological market is invaded of a great variety of wearable devices like smart watch or glasses that do not use their full potential in IoT. These devices can only acquire a litter bit part of our Bio-Information. It could be used to monitor our vital sign and create other application to introduce the human into the IoT. For example, The multifunctional suit can be used to monitor patients in their own home or nursing. The future expectations are that all people are connected to the cloud, monitoring their vital signs so that diseases can be detected through clothing.

This multifunctional suit allows to acquire ECG signal, breathing frequency, body temperature and heart rate.

Keywords: Monitor, ECG signals.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible, success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So, it is with gratitude that we acknowledge all those whose guidance and encouragement served as beacon of light and crowned our effort with success.

We would like to thank **Dr. Sanjay Jain**, Principal, CMRIT, Bangalore, for providing an excellent academic environment in the college and his never-ending support for the B.E program.

We would like to express our gratitude towards **Dr. M. Farida Begam**, Professor and HOD, Department of Information Science and Engineering CMRIT, Bangalore, who provided guidance and gave valuable suggestions regarding the project.

We consider it a privilege and honour to express our sincere gratitude to our internal guide **Dr. S. Saravanan**, Assoc. Professor, Department of Information Science & Engineering, CMRIT, Bangalore for their valuable guidance throughout the tenure of this project work.

We would also like to thank all the faculty members who have always been very Co-operative and generous. Conclusively, we also thank all the non-teaching staff and all others who have done immense help directly or indirectly during our project.

ADARSH SHARMA
KARTIK KUMAR
KARTIK KASYAP
SHASHWAT KUMAR

TABLE OF CONTENTS

Title	Page No.
Abstract.....	i
Acknowledgement.....	ii
List of Figures.....	vii
List of Tables.....	ix
CHAPTER 1	1-5
PREAMBLE.....	1
1.1 Introduction.....	1
1.2 Existing System.....	2
1.2.1 Drawbacks.....	2
1.3 Proposed System.....	3
1.4 Plan of Implementation.....	3
1.5 Problem Statement.....	4
1.6 Objective of the Project.....	5
CHAPTER 2	6-7
LITERATURE SURVEY.....	6
CHAPTER 3	8-22
THEORETICAL BACKGROUND.....	8
3.1 Internet of Things.....	8
3.1.1 Sensors: Collecting and Sending.....	9
3.1.2 Microcontroller: Arduino Uno.....	11
3.1.3 Wi-Fi Module.....	12
3.2 Model Based Design (MBD).....	13
3.2.1 Existing Technique.....	13
3.2.2 MBD Concept.....	16
3.2.3 Challenges of Model Based Design.....	17

3.3 Android Studio and Java.....	18
3.3.1 Advantages of java in android development.....	19
3.4 Arduino IDE and C++.....	20
3.5 Firebase – Cloud Storage.....	21
CHAPTER 4	23-27
SYSTEM REQUIREMENTS AND SPECIFICATION.....	23
4.1 Function Requirement.....	24
4.2 Non-Functional Requirement.....	24
4.2.1 Performance Requirement.....	25
4.2.2 Design Requirement.....	25
4.2.3 Security Constrains.....	25
4.2.4 Basic Operational Requirement.....	25
4.3 System Component and Requirements.....	26
CHAPTER 5	28-30
SYSTEM ANALYSIS.....	28
5.1 Feasibility Study.....	28
5.1.1 Economical Feasibility.....	28
5.1.2 Technical Analysis.....	29
5.1.3 Social Feasibility.....	29
5.2 Analysis.....	29
5.2.1 Performance analysis.....	29
5.2.2 Technical analysis.....	30
5.2.3 Economical analysis.....	30
CHAPTER 6	31-43
SYSTEM DESIGN.....	31
6.1 System Development and Methodology.....	31
6.1.1 Model phase.....	32
6.1.2 Advantages of Waterfall Model.....	33
6.2 Design using UML.....	33
6.3 Data Flow Diagram.....	38

6.4 Component Diagram.....	38
6.5 User Case Diagram.....	39
6.6 Sequence Diagram.....	40
6.7 Class Diagram.....	42
CHAPTER 7	44-57
IMPLEMENTATION.....	44
7.1 Android application development.....	44
7.1.1 Main activity code fragment in XML.....	44
7.1.2 Java code fragment for activity.....	47
7.1.3 Firebase connection and access data	48
7.2 Web Application development.....	50
7.2.1 Main page.....	50
7.2.2 ECG graph implementation.....	52
7.2.3 Firebase connection and retrieving data.....	55
7.3 Arduino code implementation.....	56
7.3.1 Arduino code implementation.....	56
7.3.2 Getting data from server.....	56
CHAPTER 8	58-63
TESTING.....	58
8.1 Testing Methodologies.....	58
8.1.1 White Box Testing.....	58
8.1.1.1 Advantages of White Box Testing.....	59
8.1.1.2 Disadvantages of White Box Testing.....	59
8.1.2 Black Box Testing.....	60
8.1.2.1 Advantages of Black Box Testing.....	60
8.1.2.2 Disadvantages of Black Box Testing.....	61
8.2 Unit Testing.....	61
8.3 System Testing.....	61
8.4 Quality Assurance.....	61
8.4.1 Quality Factor.....	62
8.5 Functional Test.....	62

CHAPTER 9	64-68
RESULTS AND PERFORMANCE ANALYSIS.....	64
CHAPTER 10	70-70
CONCLUSION AND FUTURE SCOPE.....	70
10.1 Future Scope.....	70
10.2 Conclusion.....	70

LIST OF FIGURES

Figure No.	Title	Page No.
Fig 1.1	Implementation Model	04
Fig 3.1.1a	ECG Sensor	09
Fig 3.1.1b	Temperature and Humidity Sensor	10
Fig 3.1.1c	Temperature Sensor	11
Fig 3.1.1d	Pulse Sensor	11
Fig 3.1.2	Arduino Duo	12
Fig 3.1.3	Wifi Module	13
Fig 3.2.1a	Extraction of Textual Format	15
Fig 3.2.1b	Explanation of Specification	15
Fig 6.1	Waterfall Mode	33
Fig 6.2	Data Flow Diagram	37
Fig 6.4	Component Diagram	38
Fig 6.5	Use Case Diagram	40
Fig 6.6	Sequence Diagram	41
Fig 8.1.1	White Box Testing	59
Fig 8.1.2	Black Box Testing	60
Fig 9.1	Circuit Diagram	64
Fig 9.2	Cloud Storage-Database	64
Fig 9.3	Cloud Storage-Authentication rule	65
Fig 9.4a	Android Splash Screen	65
Fig 9.4b	Android Main Screen1	66
Fig 9.4c	Android Main Screen 2	66
Fig 9.4d	Android Call Screen	66
Fig 9.4e	Android Map Screen	66
Fig 9.5a	Web Screen1	67

Fig 9.5b	Web Main Screen	67
Fig 9.5c	Web ECG Screen	68
Fig 9.6	ECG Graph	68

LIST OF TABLES

Table No.	Title	Page No.
Table 1.5	Problem statement with solution.....	04
Table 6.1	Symbols used in UML.....	36

Chapter 1

PREAMBLE

1.1 Introduction

Remote health monitoring has the potential to improve the quality of health services and to reduce the total cost in healthcare by avoiding unnecessary hospitalizations and ensuring that those who need urgent care get it sooner. In conjunction with cost-effective telemedicine platforms, remote health monitoring can significantly contribute to the enhancement of disease prevention, early diagnosis, disease management, treatment and home rehabilitation.

Progress in science and technology offers, for the first time, intelligence, speed, miniaturization, sophistication and new materials at low cost. In this new landscape, micro technologies, information technologies and telecommunications are a key factor. Micro sensors: Micro technologies offer the possibility of small size, but also intelligent, active device, working with low energy, wireless and non-invasive or mini invasive.

A BSN (Body Sensor Network) is a special purpose network designed to operate autonomously to connect to various medical sensors and implants located inside and outside of the human body. Introducing it in medical monitoring will offer flexibility of operation and cost saving options to both healthcare professionals and patients. They reduce user discomfort and enhance mobility. Applications in this category include monitoring of the human physiological data, tracking and monitoring of the patients inside a hospital, drug administration in hospitals etc. Vitals signs are used to measure body's basic functions, which can be helpful for monitoring general health of a person.

1.2 Existing System:

Temperature sensor is used to monitor body temperature. The photo plethysmography sensor uses the absorption of light to calculate the heart rate. ECG electrodes are used to measure the electrical activity of the heart. These above sensors are used individually accordingly. There is no such IoT system which communicates with both Android and web-based. The generic storage system was used. There was no data security and was vulnerable.

1.2.1 Drawbacks

1. The existing system was used individually, which gives data to the application stand-alone.
2. These data are not well coordinated which causes to provide wrong analysis.
3. The ECG modeling of graph is not so quite accurate
4. No visualization data was provided
5. The cost for the proposed system was high.

1.3 Proposed System

My application is based on this kind of devices to connect the human body to the cloud. The technological market is invaded of a great variety of wearable devices like smart watch or glasses that do not use their full potential in IoT. These devices can only acquire a litter bit part of our Bio-Information. It could be used to monitor our vital sign and create other application to introduce the human into the IoT. For example, The Bio-Smart Wear can be used to monitor patients in their own home or nursing. The future expectations are that all people are connected to the cloud, monitoring their vital signs so that diseases can be detected through clothing. Plan of Implementation.

1.4 Plan of Implementation

1. The project can be broken down into 4 Major states.
2. The first stage of project is gather the requirement, which includes both the hardware requirement and software requirement.
3. The second stage is develop the user interface for the application and connection from the cloud storage. The development of user interface for both android and desktop website.
4. The third stage is to assemble the hardware required for the project, form the circuit for the project, then setup the code on the hardware, connect to the server for data transfer, and receive.
5. The Final stage is to present the data captured by the sensor to the application in particular required format.

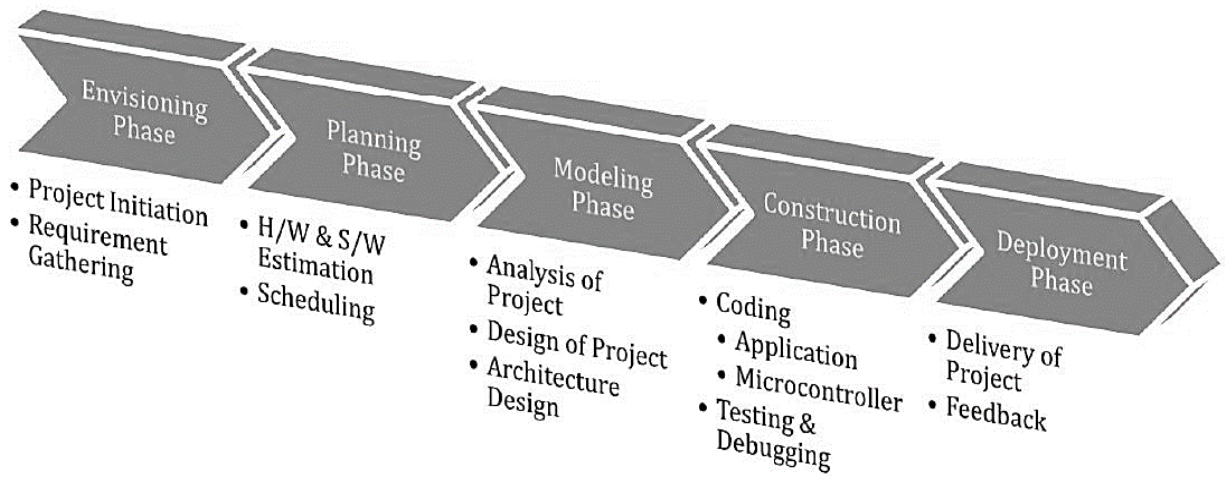


Figure 1.1: Implementation model

1.5 Problem Statement:

Problem	To leverage the usages of temperature, photo plethysmography and pulse sensor to build a IoT enabled multifunctional suit.
Affects	Initially it require the high installation cost and purchase coast but in long run is more effective and efficient.
Impact	The field of applications is very large, e.g. continuous monitoring on elderly population, professional and military activities, athletes performance and condition, and people with disabilities.
Solution	Improving the relevancy of each sensor and increasing the number of sensors for having a more global synthetic and robust information.

Table 1.1: Problem Statement with solution

1.6 Objective of the Project:

My application is based on the use of this kind of devices to connect our body to the cloud. To leverage the usages of temperature, photo plethysmography and pulse sensor to build a IoT enabled multifunctional suit. The field of applications is very large, e.g. continuous monitoring on elderly population, professional and military activities, athlete's performance and condition, and people with disabilities. Improving the relevancy of each sensor and increasing the number of sensors for having a more global synthetic and robust information.

The objective of the “IoT enabled multifunction suit” can be stated as follow:

1. Capture the data from the sensor.
2. Then send the data to the server
3. Then capture the data in the application developed perform report managing.
4. Perform action based on the data.

Chapter 2

LITERATURE SURVEY

In order to get required knowledge about various concepts related to the present application, existing literature were studied. Some of the important conclusions were made through those are listed below.

- 1.* In reference[1] paper, present a novel sensor device, namely, Smart Insole, to tackle the challenge of efficient gait monitoring in real life. The limitation and improvement, wearability and usability , further work, and healthcare-related potential applications are discussed.
- 2.* In reference[2] paper, the research is oriented toward two complementary directions: improving the relevancy of each sensor and increasing the number of sensors for having a more global synthetic and robust information.
- 3.* In reference[3] paper, it describe and discuss the wearable system future scope and further development. In addition, cutting edge development combining functional clothing and integrated electronics open a new research area and possibilities for body sensing and communicating health parameters. This paper review the current status in research and development of smart wearable health systems and applications and analyze the outstanding issues and future challenges.
- 4.* In reference[4] paper, it discuss the implementation point of the remote monitoring. In this paper a real-time heart disease monitoring system is introduced. In this paper, IoT is becoming a major platform for many services and application. Paper proposed a generic health monitoring system. The system is implemented using pulse rate sensor, Arduino UNO and ThingSpeak cloud.

5. In reference[5] paper, it presented the impact of the hybrid platform of internet of things and how the cloud computing provide back support to it. It presented the various challenges implementation of the system and the opportunity in the health care system.

6. In reference[6] paper, this paper presented the data collection system and storing of data from ambulatory patient at high resolution and send low resolution data to a remote server via wireless network. A two-tiered analysis scheme is used, where the first tier is more sensitive and less specific than the second tier.

7. In reference[7] paper, this paper presented the survey analysis on reliability , fault tolerance and technologies coexistence. This describes the wireless communication interaction of data obtained from various sensors. These data, how get securely transferred, how secure the cloud is and the fault tolerance capacity if the network.

Chapter 3

THEORETICAL BACKGROUND

Theoretical background highlighting some topics related to the project work is given below. The description contains several topics which are worth to discuss and also highlight some of their limitation that encourage going on finding solution as well as highlights some of their advantages for which reason these topics and their features are used in this project.

3.1 Internet of Things

IoT is a technology transition in which device will allow to sense and control the physical world by making objects smarter and connecting them through an intelligent network. The basic premise and goal of IoT is to “connect the unconnected”. When objects and machines can be sensed and controlled remotely across a network, a tighter integration between the physical world and computer is enabled.

In the Internet of Things, all the things that are being connected to the internet can be put into three categories.

1. Things that collect information and then send it: Sensors
2. Things that receives information and then act on it: Receivers
3. Things that do both.

3.1.1 Sensors: Collecting and Sending Information.

Sensors could be temperature sensors, motion sensor, moisture sensors, air quality sensors etc. These sensors, along with a connection, allow us to automatically collect information from the environment which, in turn, allow us to make more intelligent decisions.

Sensors used in this project are:

3.1.1.a ECG monitoring with AD8232 ECG Sensor: This Analog Heart Rate Sensor (ECG) supplies a clear measurement of the electrical pulse of the heart. It can be used to measure your heart bpm. The recommended input voltage is 5V.



Fig. 3.1.1.a ECG Sensor

3.1.1.b AM2320 Digital temperature and Humidity Sensor: Simple I2C Digital Humidity and temperature sensor, 3% humidity accuracy and 0.5°C temperature accuracy. Temperature and humidity combined sensor AM2320 digital temperature and humidity sensor is a digital signal output has been calibrated. Using special temperature and humidity acquisition technology, ensure that the product has a very high reliability and excellent long-term stability. Sensor consists of a capacitive moisture element and an integrated high-precision temperature measurement devices, and connected with a high-performance microprocessor. AM2320 communication using a single bus, two communication modes standard I2C. Standard single-

bus interface, the system integration becomes easy and quick. Ultra-small size, low power consumption, signal transmission distance up to 20 meters, making all kinds of applications and even the most demanding applications the best choice. I2C communication using standard communication sequence, the user can directly linked to the I2C communication bus without additional wiring, simple to use

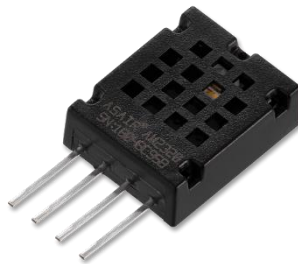


Fig. 3.1.1.b Temp. and Humidity Sensor

3.1.1.c DS18B20 1 Wire Temperature Sensor: The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line (“parasite power”), eliminating the need for an external power supply. Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

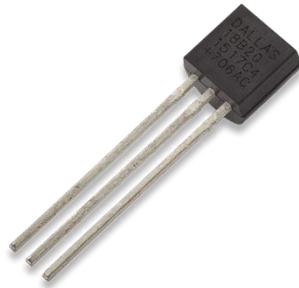


Fig 3.1.1.c Temperature Sensor

3.1.1.d Heart Rate Pulse Sensor: An analog heart pulse rate sensor. It can be used to measure your heart bpm. Its returned values is 0 to 1023 analog reading. The sensor clips onto a figure tip or ealobes and plug right into Arduino with some jumper cable.



Fig.3.1.1.d Pulse Sensor

3.1.2 Microcontroller: Arduino Uno

A **microcontroller** is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical **microcontroller** includes a processor, memory and input/output (I/O) peripherals on a single chip.

Arduino Uno is a microcontroller board based on the ATmega328P . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. Arduino is a software company, project , and user community that designs and manufactures computer open-source hardware, open-source software, and microcontroller-based kits for building digital devices and interactive objects that can sense and control physical devices. These systems provide sets of digital and analog I/O pins that can interface to various expansion boards and other circuits. The board feature serial communication interfaces including USB ,for loading programs from personal computer.

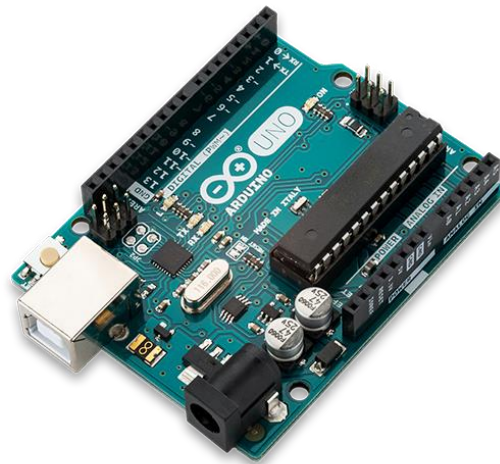


Fig. 3.1.2 Arduino Uno

3.1.3 Wifi Modules: ESP8266-01

The **ESP8266** is a low-cost Wi-Fi board, that you can easily wire to a microcontroller, and connect any project you build to the internet (of things). The microcontroller communicates with the module using AT commands, either via Software Serial or Hardware Serial. **Returned Values** is from 0 (completely dry) to 1023 (completely moist). (air/soil humidity ambient conditions) and **Requirements** Soldering of jumper/regular wires to the sensor board, to fit onto a breadboard.

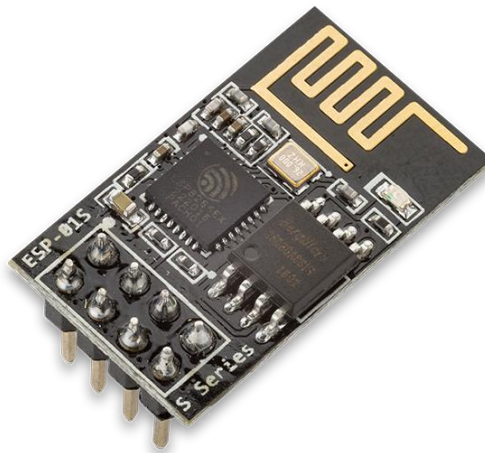


Fig. 3.1.3 Wifi Module

3.2 Model Based Design (MBD)

Model-Based Design is a process that enables fast and cost-effective development of dynamic systems, including control systems, signal processing, and communications systems.

3.2.1 Existing Technique

Specification-based Development/Behavior-Driven Development is traditionally a more manual, documentation led, method leading to manual coding. Many years of experience and development using this process lifecycle in industry have proved its worth.

Behavior-driven development (BDD) is a software development process that emerged

from test-driven development (TDD). Behavior-driven development combines the general techniques and principles of TDD with ideas from domain-driven design and object oriented analysis and design to provide software development and management teams with shared tools and a shared process to collaborate on software development.

BDD is largely facilitated through the use of a simple domain-specific language (DSL) using natural language constructs that can express the behavior and the expected outcomes. Test scripts have long been a popular application of DSLs with varying degrees of sophistication.

BDD is considered as an effective technical practice especially when the "problem space" of the business problem to solve is complex.

A template for a textual format was followed in different BDD software tools which gave a narrative story about the acceptance scenarios. The scenarios are ideally phrased declaratively rather than imperatively in the business language, with no reference to elements of the user interface through which the interactions take place. A separate subcategory of behaviordriven development is formed by tools that use specifications as an input language rather than user stories. Specification tools don't use user stories as an input format for test scenarios but rather use functional specifications for units that are being tested. These specifications often have a more technical nature than user stories and are usually less convenient for communication with business personnel than are user stories.

Story: Returns go to stock

In order to keep track of stock

As a store owner

I want to add items back to stock when they're returned.

Scenario 1: Refunded items should be returned to stock

Given that a customer previously bought a black sweater from me

And I have three black sweaters in stock.

When he returns the black sweater for a refund

Then I should have four black sweaters in stock.

Scenario 2: Replaced items should be returned to stock

Given that a customer previously bought a blue garment from me

And I have two blue garments in stock

And three black garments in stock.

When he returns the blue garment for a replacement in black

Then I should have three blue garments in stock

And two black garments in stock.

Fig 3.2.1a Example for Textual Format

Specification: Stack

When a new stack is created

Then it is empty

When an element is added to the stack

Then that element is at the top of the stack

When a stack has N elements

And element E is on top of the stack

Then a pop operation returns E

And the new size of the stack is N-1

Fig 3.2.1b Example of specification

Such a specification may exactly specify the behavior of the component being tested, but is less meaningful to a business user. As a result, specification-based testing is seen in BDD practice as a complement to story-based testing and operates at a lower level. Specification testing is often seen as a replacement for free-format unit testing.

3.2.2 MBD Concept

Model-Based Design is transforming the way engineers and scientists work by moving design tasks from the lab and field to the desktop. When software and hardware implementation requirements are included, such as fixed-point and timing behaviour, you can automatically generate code for embedded deployment and create test benches for system verification, saving time and avoiding the introduction of manually coded errors. Use Model-Based Design with Android and web to improve product quality and reduce development time by 50% or more.

In Model-Based Design, a system model is at the centre of the development process, from requirements development through design, implementation, and testing. The model is an executable specification that you continually refine throughout the development process. After model development, simulation shows whether the model works correctly. Modelbased design provides an efficient approach for establishing a common framework for communication throughout the design process while supporting the development cycle. When software and hardware implementation requirements are included with the model, such as fixed-point and timing behaviour, you can generate code for embedded deployment and create test benches for system verification, saving time and avoiding manually coded errors.

Model-Based Design allows you to improve efficiency by:

- Using a common design environment across project teams

- Linking designs directly to requirements
- Integrating testing with design to continuously identify and correct errors
- Refining algorithms through multi-domain simulation
- Generating embedded software code
- Developing and reusing test suites
- Generating documentation
- Reusing designs to deploy systems across multiple processors and hardware targets

3.2.3 Challenges of Model Based Design

What MBD suggests is essentially a role transition of software models from documentation to development. This implies an enhanced requirement on software models for completeness and precision, compared with the traditional use of models. It also demands an efficient mechanism of model-implementation mapping, which is not only about generating model-prescribed code, but also about managing the consistency between model and code over the passage of time. In general, no MBD approach can survive in the long run if the cost of model-implementation mapping significantly exceeds that of working on code directly. This section describes the research challenges in multi-aspect modeling of MBD from the perspectives of what it is, why it is hard, and how existing mechanisms are deficient in addressing it.

Multi-Aspect Modeling Software models in the development of complex software often need to describe the system from multiple aspects, such as structure, behavior, and nonfunctional properties. Important research progress has been made in this area. However, most of existing modeling technologies are based on the assumption that software models are documentation artifacts that are peripheral to code development. With regard to structure, models such as UML class diagrams may be fine for use in MBD. With regard to behavior, few models created with current technologies are amenable to software synthesis in MBD; the situation with regard to non-functional models is even worse.

The challenge is that software models in MBD not only have to contain enough details to generate relatively complete code, but also need to be, and stay, simpler than the software programs created during this process. Existing behavioral modeling methods include those that are based on formal notations and those that are more informal, but with a practical bias. None, however, provides an appropriate form for MBD. Formal behavioral modeling methods include the use of process algebras and the pi-calculus. Providing a basis for automatic analysis is one of their main purposes. They are seldom appropriate for software development because of their limited expressiveness. In most cases, developers would rather write code directly.

Examples of more informal methods include interaction diagrams, state diagrams, and activity diagrams of UML. Traditionally, these methods are mainly for communication and system comprehension. Their incompleteness properties have decided that they cannot be used alone for behavioral modeling in MDD, which emphasizes complete modeling. In cases where only executions of significance are concerned, such as architecture-centric development, practical methods like sequence diagrams may be a good choice after some form of extension.

3.3 Android Studio and Java

Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. The primary support language in android application development is java.

Java: Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible. It is intended to let application developers *write once, run anywhere* (WORA),^[17] meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.^[18] Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture.

The runtime provides an implementation of the standard java library and some mechanism for

taking the dex bytecode that apps are deployed to the device in and converting it to the machine code that actually runs on the hardware. The standard lib implementation has changed from Apache Harmony to openJDK as of Android Nougat. The runtimes prior reliance on Apache Harmony was the reason why we Android developers were deprived of newer language features like the streams API, lamdas, method references etc.

Variables: A **variable** is assigned with a data type. **Variable** is a name of memory location. There are three types of **variables** in **java**: local, instance and static. There are two types of data types in **Java**: primitive and non-primitive. Variables are defined using the assignment operator.

Function: A Java method is a collection of statements that are grouped together to perform an operation. When you call the System.out.println() method, for example, the system actually executes several statements in order to display a message on the console. Valid function names begin with an alphabetic character, and can contain letters, numbers, or underscores

Classes and Object-Oriented Programming: **Classes** and **Objects** are basic concepts of Object Oriented Programming which revolve around the real life entities. A **class** is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type.

3.3.1 Advantages of using java in Android application development.

List down are some of the key advantages of Java Programming:

- **Object Oriented** – In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform Independent** – Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

- **Simple** – Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
- **Secure** – With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architecture-neutral** – Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
- **Portable** – Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
- **Robust** – Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.

3.4 Arduino IDE and C++

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. It is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to **Arduino** compatible boards, but also, with the help of 3rd party cores, other vendor development boards.

C++: The language used to program the Arduino is C++. C++ is a superset of C, adding classes and changing the behavior of struts in subtle ways. C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.

C++ is being heavily used in developing Medical and Engineering Applications like software's for MRI machines, high-end CAD/CAM systems etc. The Arduino IDE is actually just a thin wrapper around GCC and the language using fact C/C++. The advantage of using Arduino is that

is has a core set of library functions that control the peripheral hardware of whatever microcontroller you are using, and these functions are standard. So can use the same program on an AVR or an ARM etc., with only minor tweaks.

3.5 Firebase: Real time data cloud storage

Firestore is a modern technology that allows for storing and managing various media content generated by mobile app users. ... **Firestore** SDK also uses a declarative data protection language, which allows you to manage the access to the files in the **Firestore** and make them public or private. Firestore allows you to smoothly switch from an app prototype to a full-fledged product thanks to being highly scalable and the ability to process Exabyte of data.

3.6 Web Application : Front End

Front-end web development is the practice of converting data to a graphical interface, through the use of HTML, CSS, and JavaScript, so that users can view and interact with that data.

HTML: HTML (Hypertext Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content. HTML uses "markup" to annotate text, images, and other content for display in a Web browser. An HTML element is set off from other text in a document by "tags", which consist of the element name surrounded by "<" and ">". The name of an element inside a tag is case insensitive. That is, it can be written in uppercase, lowercase, or a mixture.

CSS: Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts.

Vue.js: Vue.js is an open-source model–view–view model JavaScript framework for building user interfaces and single-page applications. Vue.js aims to be an approachable, versatile, performant, maintainable, and testable JavaScript framework. Vue also aims to be progressive, meaning that if you have an existing application you can use Vue in just one part of your front-

end that needs a more interactive experience.

Chapter 4

SYSTEM REQUIREMENT SPECIFICATION

A System Requirement Specification (SRS) is basically an organization's understanding of a customer or potential client's system requirements and dependencies at a particular point prior to any actual design or development work. The information gathered during the analysis is translated into a document that defines a set of requirements. It gives the brief description of the services that the system should provide and also the constraints under which, the system should operate. Generally, SRS is a document that completely describes what the proposed software should do without describing how the software will do it. It's a two-way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time.

SRS document itself states in precise and explicit language those functions and capabilities a software system (i.e., a software application, an ecommerce website and so on) must provide, as well as states any required constraints by which the system must abide. SRS also functions as a blueprint for completing a project with as little cost growth as possible. SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it.

Requirement is a condition or capability to which the system must conform. Requirement Management is a systematic approach towards eliciting, organizing and documenting the requirements of the system clearly along with the applicable attributes. The elusive difficulties of requirements are not always obvious and can come from any number of sources.

4.1 Functional Requirement

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. In this system following are the functional requirements:-

Following are the functional requirements on the system:

1. The whole process can be handled at minimal human interaction with android and web both.
2. The application automatically receive the captured data from server.
3. The user can call emergency , map location and ECG graph on demand
4. The system gives a warning message.

4.2 Non Functional Requirement

Non functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Non functional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as:-

- Performance Requirements
- Design Requirements
- Security constraints
- Basic Operational Requirement

4.2.1 Performance Requirements

The resulting device should perform on human body and various sensors having the ability of compatibility more than the ones in the market. Additionally a more general communication protocol and wireless controller hardware should be used to make the system perform on much more devices. The system should access data in reasonable time. The data transfers between the devices such as actuators and sensors with master controller should not exceed the time limit of 3 seconds and lie under the throughput of 250 Kbps. Additionally, the system should service with the 7 days / 24 hours availability. The system should work smoothly with other existing connection networks at home.

Non functional requirements are also called the qualities of a system. These qualities can be divided into execution quality & evolution quality. Execution qualities are security & usability of the system which are observed during run time, whereas evolution quality involves testability, maintainability, extensibility or scalability.

4.2.2 Design Requirements

The reporting of the project should be in IEEE standards and its diagrams should be drawn in UML standards. The interface between the system components should be well described to make the user control easier. As an environment constraint, master controller software should be developed on Linux system. Moreover, there is another constraint on wireless communication protocol. In this system, ZigBee wireless protocol should be used to make the devices communicate. The transmitted information between the devices should be carried in encrypted form, as a security constraint.

4.2.3 Security constraints

Security is another design constraint. The system should encrypt/decrypt the data transmitted between the Xbee-Pro devices.

4.2.4 Basic Operational Requirements

The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, will be related to these following points:-

Mission profile or scenario: It describes about the procedures used to accomplish mission objective. It also finds out the effectiveness or efficiency of the system.

Performance and related parameters: It points out the critical system parameters to accomplish the mission.

Utilization environments: It gives a brief outline of system usage. Finds out appropriate environments for effective system operation.

Operational life cycle: It defines the system lifetime.

4.3 System components and requirement

H/W components:

1. Arduino 101& Genuino 101
2. SparkFun Single lead heart rate monitor – AD8232
3. DS 18B20 1 wire temperature sensor
4. Pulse sensor
5. 9V battery
6. Wifi Module
7. Logic level Converter
8. 1 LED
9. Breadboard

Software apps and online services:

1. Arduino IDE
2. Android Studio
3. Google Storage-Firebase
4. JASON
5. Web development editor

Hand Tools and fabrication machines:

1. Sports T shirt
2. Soldering iron(generic)
3. ECG Electrode

Chapter 5

SYSTEM ANALYSIS

Analysis is the process of finding the best solution to the problem. System analysis is the process by which we learn about the existing problems, define objects and requirements and evaluates the solutions. It is the way of thinking about the organization and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis which gives the target for design and development.

5.1 Feasibility Study

All systems are feasible when provided with unlimited resource and infinite time. But unfortunately this condition does not prevail in practical world. So it is both necessary and prudent to evaluate the feasibility of the system at the earliest possible time. Months or years of effort, thousands of rupees and untold professional embarrassment can be averted if an ill-conceived system is recognized early in the definition phase. Feasibility & risk analysis are related in many ways. If project risk is great, the feasibility of producing quality software is reduced. In this case three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

5.1.1 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development

of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

5.1.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

5.1.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

5.2 Analysis

5.2.1 Performance Analysis

For the complete functionality of the project work, the project is run with the help of healthy networking environment. Performance analysis is done to find out whether the proposed system. It is essential that the process of performance analysis and definition must be conducted in parallel.

5.2.2 Technical Analysis

System is only beneficial only if it can be turned into information systems that will meet the organization's technical requirement. Simply stated this test of feasibility asks whether the system will work or not when developed & installed, whether there are any major barriers to implementation. Regarding all these issues in technical analysis there are several points to focus on:-

Changes to bring in the system: All changes should be in positive direction, there will be increased level of efficiency and better customer service.

Required skills: Platforms & tools used in this project are widely used. So the skilled manpower is readily available in the industry.

Acceptability: The structure of the system is kept feasible enough so that there should not be any problem from the user's point of view.

5.2.3 Economical Analysis

Economic analysis is performed to evaluate the development cost weighed against the ultimate income or benefits derived from the developed system. For running this system, we need not have any routers which are highly economical. So the system is economically feasible enough.

Chapter 6

SYSTEM DESIGN

Design is a meaningful engineering representation of something that is to be built. It is the most crucial phase in the developments of a system. Software design is a process through which the requirements are translated into a representation of software. Design is a place where design is fostered in software Engineering. Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. Design is the perfect way to accurately translate a customer's requirement in the finished software product. Design creates a representation or model, provides details about software data structure, architecture, interfaces and components that are necessary to implement a system. The logical system design arrived at as a result of systems analysis is converted into physical system design.

6.1 System development methodology

System development method is a process through which a product will get completed or a product gets rid from any problem. Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps which is used for product progress. The development method followed in this project is waterfall model.

6.1.1 Model phases

The waterfall model is a sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Requirement initiation, Analysis, Design, Implementation, Testing and maintenance.

Requirement Analysis: This phase is concerned about collection of requirement of the system.

This process involves generating document and requirement review.

System Design: Keeping the requirements in mind the system specifications are translated in to a software representation. In this phase the designer emphasizes on:-algorithm, data structure, software architecture etc.

Coding: In this phase programmer starts his coding in order to give a full sketch of product. In other words system specifications are only converted in to machine readable compute code.

Implementation: The implementation phase involves the actual coding or programming of the software. The output of this phase is typically the library, executables, user manuals and additional software documentation

Testing: In this phase all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation.

Maintenance: The maintenance phase is the longest phase in which the software is updated to fulfill the changing customer need, adapt to accommodate change in the external environment, correct errors and oversights previously undetected in the testing phase, enhance the efficiency of the software.

6.1.2 Advantages of the Waterfall Model

- Clear project objectives.
- Stable project requirements.
- Progress of system is measurable.
- Strict sign-off requirements.
- Helps you to be perfect.
- Logic of software development is clearly understood.
- Production of a formal specification
- Better resource allocation.

- Improves quality. The emphasis on requirements and design before writing a single line of code ensures minimal wastage of time and effort and reduces the risk of schedule slippage.
- Less human resources required as once one phase is finished those people can start working on to the next phase.

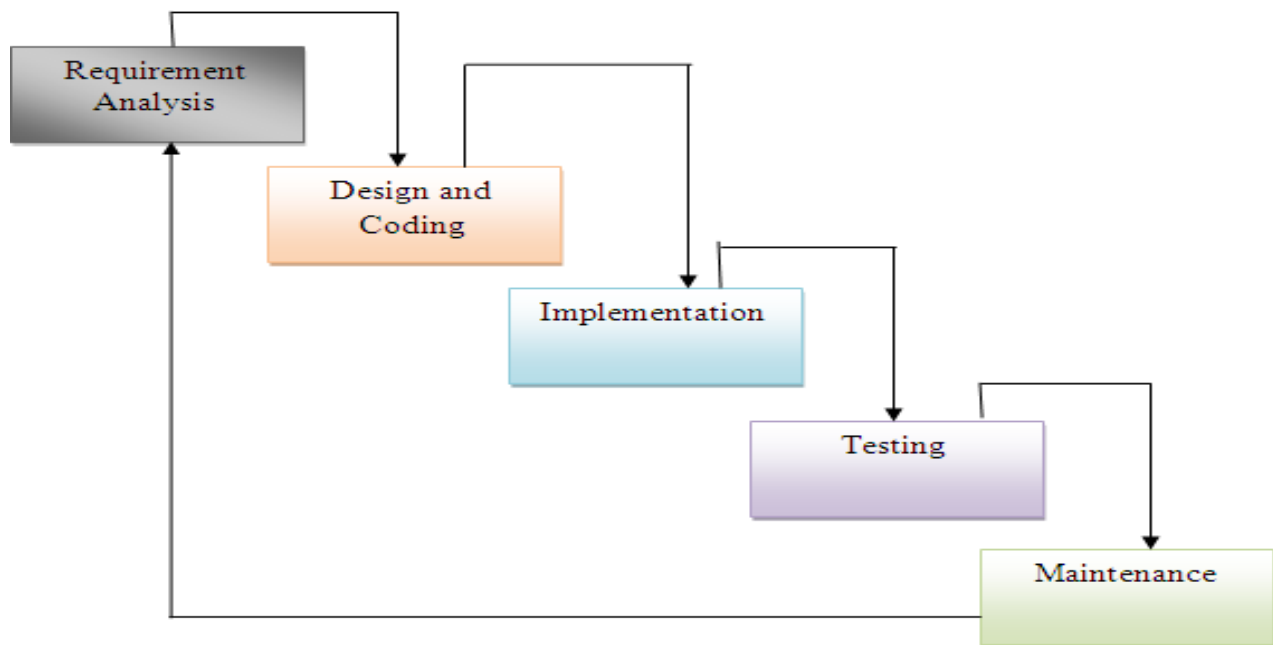
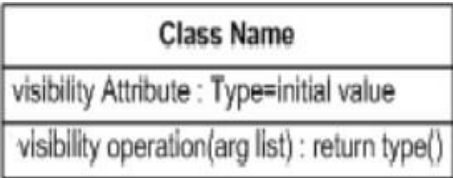
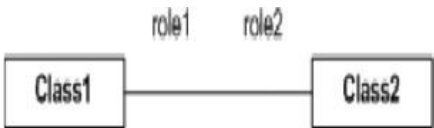


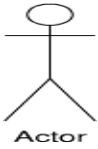



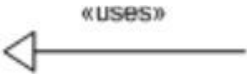




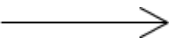
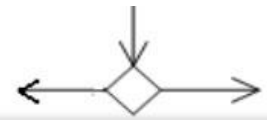
Fig 6.1: Waterfall model

6.2 Design Using UML

Designing UML diagram specifies, how the process within the system communicates along with how the objects within the process collaborate using both static as well as dynamic UML diagrams since in this ever-changing world of Object Oriented application development, it has been getting harder and harder to develop and manage high quality applications in reasonable amount of time. As a result of this challenge and the need for a universal object modeling language every one could use, the Unified Modeling Language (UML) is the Information industries version of blue print. It is a method for describing the systems architecture in detail.

Easier to build or maintains system, and to ensure that the system will hold up to the requirement changes.

Sl. No	Symbol Name	Symbol	Description
1	Class		Classes represent a collection of similar entities grouped together.
2	Association		Association represents a static relation between classes.
3	Aggregation		Aggregation is a form of association. It aggregates several classes into a single class.
4	Composition		Composition is a special type of aggregation that denotes a strong ownership between classes.
5	Actor		Actor is the user of the system that reacts with the system.

6	Use Case		A use case is an interaction between system and the external environment.
7	Relation (Uses)		It is used for additional purpose communication.
8	Communication		It is the communication between use cases.
9	State		It represents the state of process. Each state goes through various flows.
10	Initial State		It represents initial state of object.
11	Final State		It represents final state of object.
12	Control Flow		It represents decision making process for object.
13	Decision Box		It represents the decision making process from a constraint.




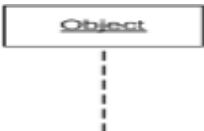
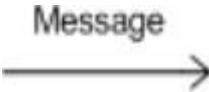
14	Data Process/ State		A circle in a DFD represents a state or process which has been triggered due to some other event or action.
15	External Entity		It represents external entity such as Keyboard, sensors, etc which are used in the system.
16	Transition		It represents any communication that occurs between processes.
17	Object Lifeline		Object lifeline represents the vertical dimension that object communicates.
18	Message		It represents messages exchanged.

Table 6.1: Symbols used in UML

6.3 Data Flow Diagram

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

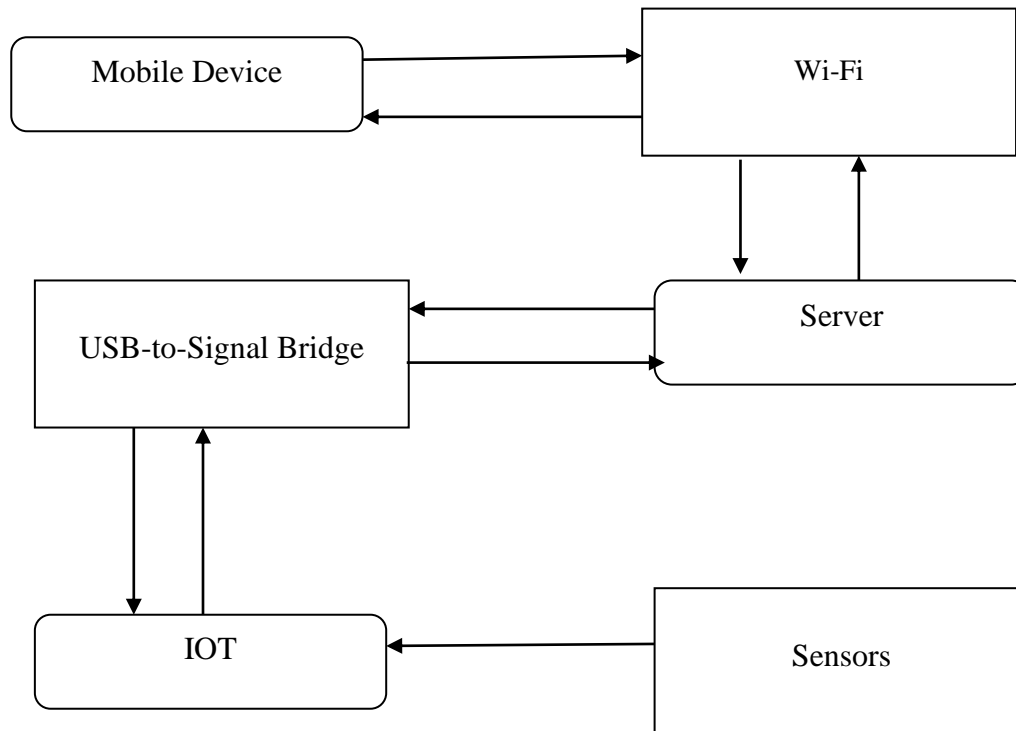


Fig 6.2 Data Flow Diagram

The data flow diagram essentially shows how the data control flows from one module to another. Unless the input filenames are correctly given the program cannot proceed to the next module. Once the correct input filenames are given by the user parsing is done individually for each file. The required information is taken in parsing and an adjacency matrix is generated for that. From the adjacency matrix, a lookup table is generated giving paths for blocks. And the final sequence is computed with the lookup table and the final required code is generated in an output file. In case of multiple file inputs, the code for each is generated and combined together.

6.4 COMPONENT DIAGRAM

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

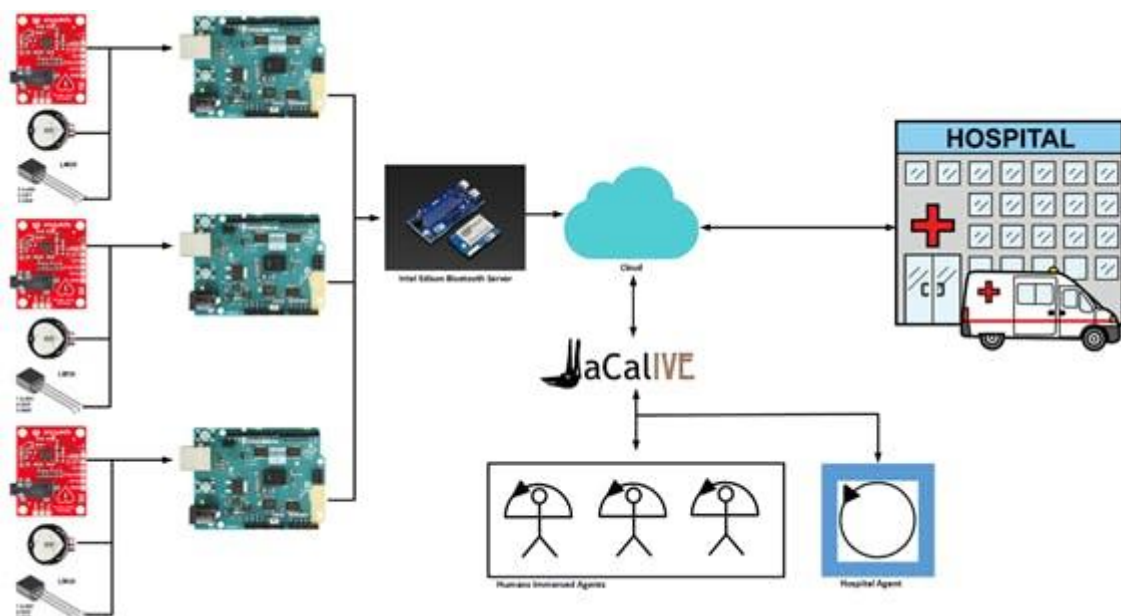


Fig 6.4 Component Diagram

The component diagram for the decentralized system ideally consists of different modules that are represented together via a common module for the user. The user is required to have the input files in the current folder where the application is being used.

It is interesting to note that all the sequence of activities that are taking place are via this module itself, i.e. the parsing and the process of computing the final sequence. The parsing redirects across the other modules till the final code is generated.

6.5 Use case Diagram:

A use case defines a goal-oriented set of interactions between external entities and the system under consideration. The external entities which interact with the system are its actors. A set of use cases describe the complete functionality of the system at a particular level of detail and it can be graphically denoted by the use case diagram.

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

In software and systems engineering, a use case is a list of steps, typically defining interactions between a role (known in Unified Modeling Language (UML) as an "actor") and a system, to achieve a goal. The actor can be a human, an external system, or time.

In systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals. The detailed requirements may then be captured in Systems Modeling Language (SysML) or as contractual statements.

The Sequence of activities that are carried out are the same as the other diagrams .Use case for this module indicates the users interaction with the system as a whole rather than individual modules .All the encryption mechanisms are carried out via the login page that redirects the user to the particular functionality that he or she wishes to implement.

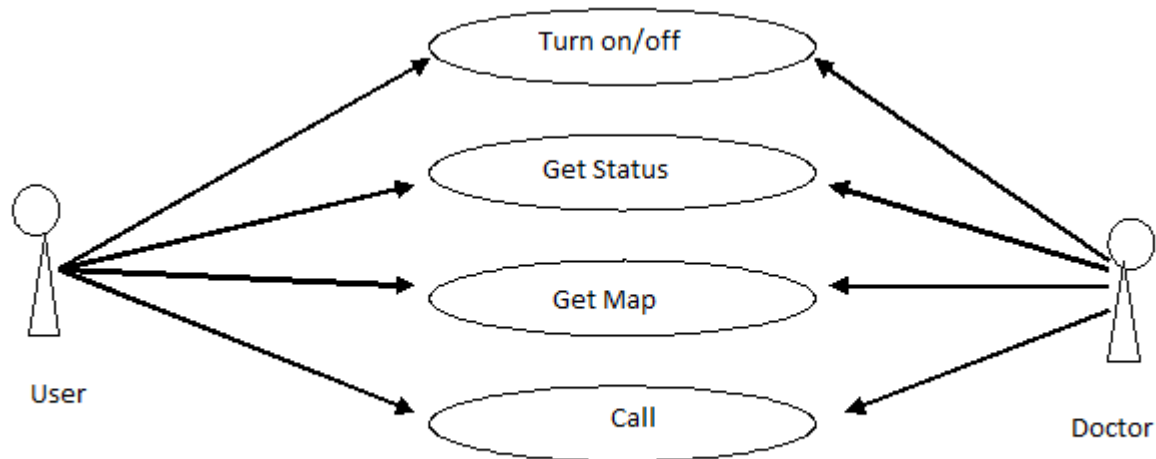


Fig. 6.5 Use case diagram

6.6 Sequence Diagram:

Sequence diagrams are an easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and the environment. A sequence diagram shows an interaction arranged in a time sequence. A sequence diagram has two dimensions: vertical dimension represents time, the horizontal dimension represents the objects existence during the interaction.

A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that

live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

The sequence diagram shows the set of events that occur once the input file has been given. The first step is to parse, where the required data will be put into a structure. Then the adjacency matrix is created which will be used when the lookup table is created and also while computing the final sequence. Once the final sequence is generated, the final code is generated which also uses the structure that was created during parsing. The user will get the generated code for the file that he has inputted.

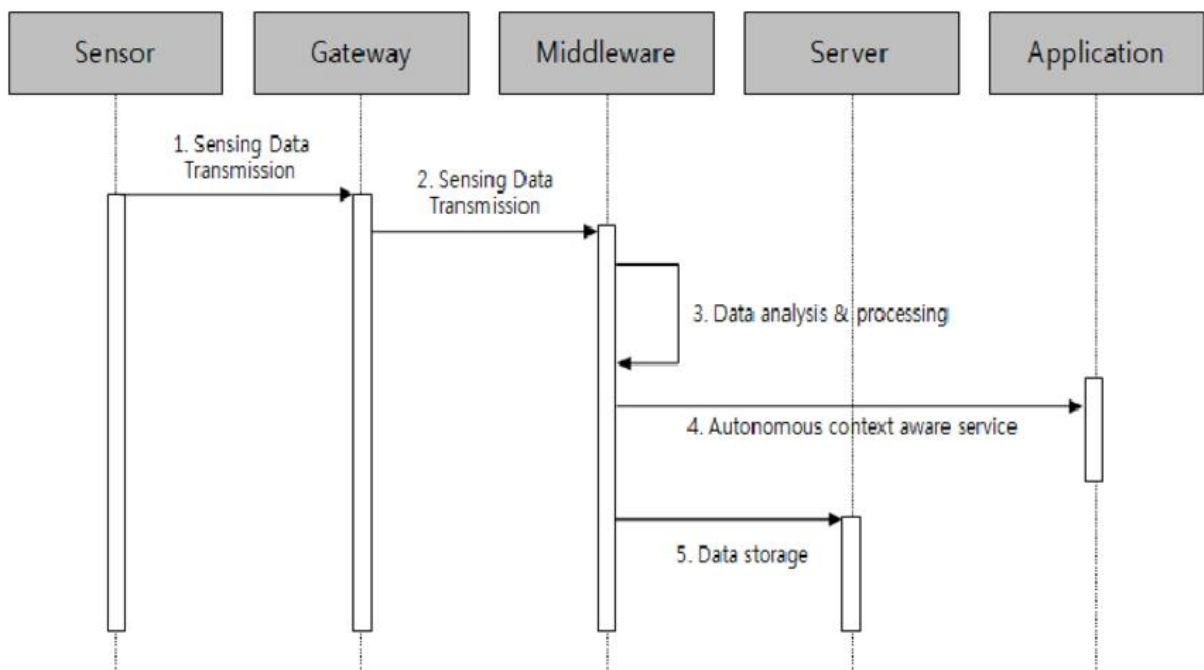


Fig 6.6 Sequence Diagram

6.7 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of object oriented modelling. It is used both for general conceptual modelling of the systematics of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

In the diagram, classes are represented with boxes which contain three parts:

- The top part contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
- The middle part contains the attributes of the class. They are left-aligned and the first letter is lowercase.
- The bottom part contains the methods the class can execute. They are also left-aligned and the first letter is lowercase.

In the design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the static relations between those objects. With detailed modelling, the classes of the conceptual design are often split into a number of subclasses.

In order to further describe the behavior of systems, these class diagrams can be complemented by a state diagram or UML state machine.

The class diagram for the decentralized system depicts different modules as classes that contain specific objects and attributes. All the classes inherit data from the parsing class and adjacency class in order to carry out the specific functions that are defined inside them.

The aggregation symbol between the lookup table class and the adjacency class indicates that the adjacency creation process must take place before the lookup table class is used. The objects that are present in the adjacency class must be used in order to generate the lookup table and the final code. The parsing class takes the filenames from the user and only the data that is necessary is taken from that file

Chapter 7

IMPLEMENTATION

Our Project divided into various stage and each stage having code section. Some important section are:

- Application Development as Android application
- Application Development as Web application
- Arduino code upload to hardware
- Firebase setup

7.1 Android Application Development

In Android application development various activity are playing role all together. Below are the important activity:

1. Main Activity Layout in XML
2. Java code for the main activity.
3. Firebase –connection and retrievals of data

7.1.1 Main Activity Layout code fragment in XML

This section include the code for the activity layout, which act as user interface in the application. This activity include various kind of layout like constrain layout, linear layout, relative layout etc. It also contain four button. Below the code for each section

Layout code : Text View

```
<TextView
    android:id="@+id/temperature"
    android:layout_width="wrap_content"
```



```
android:layout_height="wrap_content"
android:layout_centerVertical="true"
android:layout_marginLeft="25dp"
android:gravity="center_vertical"
android:text="Body Temperature" />
```

Layout Code : Image View

<ImageView

```
android:layout_width="25dp"
android:layout_height="wrap_content"
android:layout_alignParentEnd="true"
android:layout_centerVertical="true"
android:layout_gravity="center_vertical"
android:layout_marginRight="100dp"
android:gravity="center_vertical"
android:src="@drawable/thermometer" />
```

Layout Code : Button

<Button

```
android:id="@+id/mapButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="left"
android:background="#bbdefb"
android:text="MAP"
android:elevation="30dp"
android:foreground="?attr/selectableItemBackground"
/>
```

Layout code: Card View

```
<androidx.cardview.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="70dp"
    app:cardBackgroundColor="#81c784"
    >
<...>//other nested View

</androidx.cardview.widget.CardView>
```

Layout code: Main Constrain layout, Relative layout, Scroll View

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#f8bbd0"
    tools:context=".MainActivity">

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"><RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp">

<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <androidx.cardview.widget.CardView
            android:layout_width="match_parent"
            android:layout_height="70dp"
            app:cardBackgroundColor="#81c784"
```

```
        />
        //bunch of nested view, button, images

    </LinearLayout>
</ScrollView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

7.1.2 Java Code fragment for activity

Button Implementation: Call Button and Map Button

```
private Button call,map,status,ecg;private Button
    call,map,status,ecg;
call=findViewById(R.id.callButton);
call.setOnClickListener(this);
@Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.callButton:

                Toast.makeText(this,"calling...",Toast.LENGTH_SHORT).show();
                    String ph="tel:9620519930";
                    Intent callIntent = new Intent(Intent.ACTION_CALL);
                    callIntent.setData(Uri.parse(ph));
                    if
                        (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.CALL_PHONE) !=
PackageManager.PERMISSION_GRANTED) {

                    ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.CALL_PHONE},1);
                        } else startActivity(callIntent);
                    break;
```

```
        case R.id.mapButton: // map button implementation

Toast.makeText(this, "mapping...", Toast.LENGTH_SHORT).show();
        Uri gmmIntentUri = Uri.parse("geo:0,0?q=hospital");
        Intent mapIntent = new Intent(Intent.ACTION_VIEW,
gmmIntentUri);
        mapIntent.setPackage("com.google.android.apps.maps");
        startActivity(mapIntent);
        break;
// Ecg Button implemenation
        case R.id.getEcgButton:
        ...
        ...
        break;
//Getting status from server-firebase
        case R.id.getStatusButton:
        ...
        ...
        ...
        break;
    }    }
```

7.1.3 Firebase connection and access data code fragment

```
private DatabaseReference mDatabase=
    FirebaseDatabase.getInstance().getReference();
        Toast.makeText(this, "getting
status....!", Toast.LENGTH_SHORT).show();
        mDatabase.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                Map<String, Object> value = (Map<String, Object>)
```

```
dataSnapshot.getValue();

        st=value.get("sTemp").toString();
        bt=value.get("bTemp").toString();
        bb=value.get("beatRate").toString();
        Log.d("msg", ""+value);
        bTemp.setText(bt);
        eTemp.setText(st);
        heartbeat.setText(bb);
        if
(Double.valueOf(bTemp.getText().toString())>97.0){
            tempOK.setImageResource(R.drawable.wrong);
        }
        else
tempOK.setImageResource(R.drawable.feedback);
            if(Integer.valueOf(heartbeat.getText().toString())>120 ||
Integer.valueOf(heartbeat.getText().toString())<70){
heartbeatOK.setImageResource(R.drawable.wrong);
            }else
            heartbeatOK.setImageResource(R.drawable.feedback);
        }
        @Override
        public void onCancelled(DatabaseError error) {
            // Failed to read value
            Log.w("msg", "Failed to read value.",
error.toException());
        }
    });
    if (Double.valueOf(bTemp.getText().toString())>97.0){

        tempOK.setImageResource(R.drawable.wrong);
    }
}
```

```
        else tempOK.setImageResource(R.drawable.feedback);
        if(Integer.valueOf(heartbeat.getText().toString())>120 ||
Integer.valueOf(heartbeat.getText().toString())<70) {

                heartbeatOK.setImageResource(R.drawable.wrong);
        }else
        heartbeatOK.setImageResource(R.drawable.feedback);
```

7.2 Web Application Development.

In Web application development various page are playing role all together. Below are the important :

1. Main page
2. ECG graph implementation.
3. Firebase –connection and retrievals of data

7.2.1 Main Page

```
<template>
  <div class="template-container">
<!--    <h1>Welcome!</h1>-->
    <div class="mt-5"></div>
    <div class="d-flex justify-content-space-around">
      <div class="column">
        <label>BODY TEMPERATURE</label>
        <div class="demo-box">
          <v-slider
            v-model="temp"
            :marks="tempmarks"
            :min=95
            :max=110
```

```
        :tooltip="'always'"
        :disabled="true"
      >
    </v-slider></div></div>
<div class="column">
  <label>PULSE RATE</label>
  <div class="demo-box"> <v-slider
    v-model="pulse"
    :interval="10"
    :marks="pulsemarks"
    :tooltip="'always'"
    :disabled="true"
  >
  </v-slider></div>
</div>
<div class="column">
  <label>OUTSIDE TEMPERATURE</label>
  <div class="demo-box">
    <v-slider
      v-model="envtemp"
      :interval="10"
      :min=70
      :max=110
      :marks="tempmarks1"
      :tooltip="'always'"
      :disabled="true"
    >
    </v-slider></div></div></div>    <div class="mt-5"></div>
<div class="mt-5"></div>
<div class="d-flex justify-content-space-around">
  <div class="column">
```

```
<button class="button button-theme"
@click.prevent="getInstance()">Get Status</button>
</div></div> <div>    <Footer /></div>
</div></template>
```

7.2.2 ECG graph implementation

```
// eslint-disable-next-line no-unused-vars
class ECG {
  constructor () {
    /* The width ratios allows for the normalization of the widths of
the
        different segments of the pqrst wave. */
    this.PQRST_WAVE_WIDTH_RATIOS = {"p": 12,"pq": 2,"q": 2,"r":
6,"s": 3,"st": 2,"t": 12,"tp": 2}
    this.norm_array = [0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
    this.data_cursor = 0
    this.data_buffer = [] }
  getDataCursor () {return this.data_cursor}
  getStepSize () {return 1 /
(Object.keys(this.PQRST_WAVE_WIDTH_RATIOS).length *
this.norm_array.length)}
  tick () {
    if (this.data_buffer.length == 0) {
      this.data_buffer = this._generatePQRSTWave()}
    this.data_cursor += this.getStepSize()
    return this.data_buffer.shift()}

  /**
   * generate a PQRST wave and append it to the ECG data.
   */
  generatePQRSTWave () {
```



```
// P mimics a beta distribution
var p = (x) => 2 * Math.pow(x, 3) * (1 - x),
    // Q mimics the -ve part of a sine wave
    q = (x) => -1 * Math.pow(1.1, Math.sin(x, Math.PI)) + 1,
    // R mimics the +ve part of a skewed sine wave
    r = (x) => Math.pow(7, Math.sin(x, Math.PI)) - 1,
    // S mimics the -ve part of a skewed sine wave
    s = (x) => -1 * Math.pow(1.5, Math.sin(x, Math.PI)) + 1,
    // T mimics a beta distribution
    t = (x) => 5 * Math.pow(x, 2) * (1 - x),
    // pq, st, and tp segments mimic y=0
    zero_segment = (x) => 0,

    /* generate plot points for a single pqrst wave
       y points for each segment */
    p_y = this.norm_array.map(p),
    pq_y = this.norm_array.map(zero_segment),
    q_y = this.norm_array.map(q),
    r_y = this.norm_array.map(r),
    s_y = this.norm_array.map(s),
    st_y = this.norm_array.map(zero_segment),
    t_y = this.norm_array.map(t),
    tp_y = this.norm_array.map(zero_segment),

    // map normalized domain (0.0-1.0) to absolute domain
    sum_width_ratios =
extObjectValues(this.PQRST_WAVE_WIDTH_RATIOS).reduce((acc, x) => {
    return acc + x
}, 0.0),
```

```
        p_x = this.norm_array.map(x => x *
this.PQRST_WAVE_WIDTH_RATIOS.p / sum_width_ratios),
        pq_x = this.norm_array.map(x => x *
this.PQRST_WAVE_WIDTH_RATIOS.pq / sum_width_ratios),
        q_x = this.norm_array.map(x => x *
this.PQRST_WAVE_WIDTH_RATIOS.q / sum_width_ratios),
        r_x = this.norm_array.map(x => x *
this.PQRST_WAVE_WIDTH_RATIOS.r / sum_width_ratios),
        s_x = this.norm_array.map(x => x *
this.PQRST_WAVE_WIDTH_RATIOS.s / sum_width_ratios),
        st_x = this.norm_array.map(x => x *
this.PQRST_WAVE_WIDTH_RATIOS.st / sum_width_ratios),
        t_x = this.norm_array.map(x => x *
this.PQRST_WAVE_WIDTH_RATIOS.t / sum_width_ratios),
        tp_x = this.norm_array.map(x => x *
this.PQRST_WAVE_WIDTH_RATIOS.tp / sum_width_ratios),
        /* Apply the wave offset + segment offset to each segment
        i.e. t should start after r finishes, and r should start
after s, etc. */
        segment_offset = this.data_cursor
        p_x = p_x.map(x => x + segment_offset)
        segment_offset += this.PQRST_WAVE_WIDTH_RATIOS.p /
sum_width_ratios
        pq_x = pq_x.map(x => x + segment_offset)
        segment_offset += this.PQRST_WAVE_WIDTH_RATIOS.pq /
sum_width_ratios
        q_x = q_x.map(x => x + segment_offset)
        segment_offset += this.PQRST_WAVE_WIDTH_RATIOS.q /
sum_width_ratios
        r_x = r_x.map(x => x + segment_offset)
        segment_offset += this.PQRST_WAVE_WIDTH_RATIOS.r /
```

```
sum_width_ratios
    s_x = s_x.map(x => x + segment_offset)
    segment_offset += this.PQRST_WAVE_WIDTH_RATIOS.s /
sum_width_ratios
    st_x = st_x.map(x => x + segment_offset)
    segment_offset += this.PQRST_WAVE_WIDTH_RATIOS.st /
sum_width_ratios
    t_x = t_x.map(x => x + segment_offset)
    segment_offset += this.PQRST_WAVE_WIDTH_RATIOS.t / sum_width_ratios
    tp_x = tp_x.map(x => x + segment_offset)
    var x = [p_x,pq_x,q_x,r_x,s_x,st_x,t_x,tp_x].reduce((a, b) =>
a.concat(b), []),
        y = [p_y,pq_y,q_y,r_y,s_y,st_y,t_y,tp_y].reduce((a, b) =>
a.concat(b), [])
    return x.map((e, i) => [e,y[i]])}}
```

7.2.3 Firebase-connection and retrievals of data

```
"methods":
{
  getInstance () {
    this.$store.dispatch("loader", true)
    axios.get("https://formal-branch-
254305.firebaseio.com/.json", {
    }).
    then(response => {
      return response.data
    }).
    then(response => {
      // eslint-disable-next-line no-console
      console.log(response)
      this.temp = response.body_temp
      this.pulse = response.pulse rate
```

```
        this.envtemp = response.env_temp
    }).
    finally(() => {
        this.$store.dispatch("loader", false)}}}],
    created () {
        // this.getInstance()}}
```

7.3 Arduino code implementation

This section contains the important code fragment for Arduino.

7.3.1 Connection with firebase and Wi-Fi connection

```
#define WIFI_PIN_TX    11
#define WIFI_PIN_RX    10
const char *SSID      = "WIFI-SSID"; // Enter your Wi-Fi name
const char *PASSWORD = "PASSWORD" ; // Enter your Wi-Fi password
char* const host = "www.google.com";
//firebase connection
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

Firebase.getFloat("number");//get value from firebase
//update value
Firebase.setFloat("number", 43.0);
```

7.3.2 Getting data from sensor

```
// DS18B20 1-Wire Temperature Sensor - Test Code
    // Read DS18B20 temp sensor value in degrees Celsius. for
degrees Fahrenheit use ds18b20.ReadTempF()
    float ds18b20TempC = ds18b20.readTempC();

// Heart Rate Pulse Sensor - Test Code
```

```
//Measure Heart Rate
int heartpulseBPM = heartpulse.BPM;
Serial.println(heartpulseBPM);
if (heartpulse.QS == true) {
Serial.println("PULSE");
    heartpulse.QS = false;
```

Chapter 8

TESTING

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all the system elements have been properly integrated and perform allocated functions. The testing process is actually carried out to make sure that the product exactly does the same thing what is supposed to do. In the testing stage following goals are tried to achieve:-

- To affirm the quality of the project.
- To find and eliminate any residual errors from previous stages.
- To validate the software as a solution to the original problem.
- To provide operational reliability of the system.

8.1 Testing Methodologies

There are many different types of testing methods or techniques used as part of the software testing methodology. Some of the important testing methodologies are:

8.1.1 White box testing

White box testing (clear box testing, glass box testing, and transparent box testing or structural testing) uses an internal perspective of the system to design test cases based on internal structure. It requires programming skills to identify all paths through the software. The tester chooses test case inputs to exercise paths through the code and determines the appropriate outputs. While white box testing is applicable at the unit, integration and system levels of the software testing process, it is typically applied to the unit. While it normally tests paths within a unit, it can also test paths between units during integration, and between subsystems during a system level test.

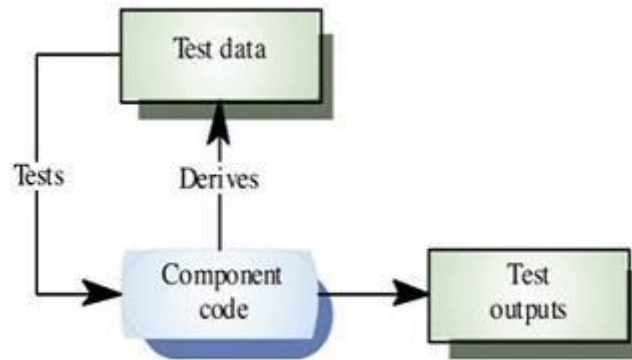


Fig 8.1.1: White Box Testing

8.1.1.1 Advantages of White Box Testing

- To start the white box testing of the desired application there is no need to wait for user face (UI) to be completed. It covers all possible paths of code which will ensure a thorough testing.
- It helps in checking coding standards.
- Tester can ask about implementation of each section, so it might be possible to remove unused/deadlines of codes helps in reducing the number of test cases to be executed during the black box testing.
- As the tester is aware of internal coding structure, then it is helpful to derive which type of input data is needed to test the software application effectively.
- White box testing allows you to help in code optimization

8.1.1.2 Disadvantages of White Box Testing

- To test the software application a highly skilled resource is required to carry out testing who has good knowledge of internal structure of the code which will increase the cost.
- Updating the test script is required if there is change in requirement too frequently.
- If the application to be tested is large in size, then exhaustive testing is impossible.

- It is not possible for testing each and every path/condition of software program, which might miss the defects in code.
- White box testing is a very expensive type of testing.
- To test each paths or conditions may require different input conditions, so in order to test full application, the tester need to create range of inputs which may be a time consuming.

8.1.2 Black box Testing

Black box testing focuses on the functional requirements of the software. It is also known as functional testing. It is a software testing technique whereby the internal workings of the item being tested are not known by the tester. For example, in a black box test on software design the tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs.

The tester does not ever examine the programming code and does not need any further knowledge of the program other than its specifications. It enables us to derive sets of inputs that will fully exercise all functional requirements for a program.

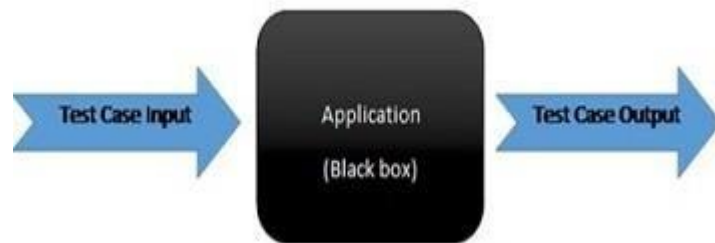


Fig 8.1.2: Black Box Testing

8.1.2.1 Advantages of Black Box Testing

- The test is unbiased as the designer and the tester are independent of each other.
- The tester does not need knowledge of any specific programming languages.
- The test is done from the point of view of the user, not the designer.

- Test cases can be designed as soon as the specifications are complete.

8.1.2.2 Disadvantages of Black Box Testing

- The test inputs need to be from large sample space. That is, from a huge set of data this will take time.
- Also it is difficult to identify all possible inputs in limited testing time. So writing test cases is slow and difficult.
- Chances are more that there will be unidentified paths during this testing.

8.2 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.3 System Testing

This information contributes towards reducing the ambiguity about the system. For example, when deciding whether to release a product, the decision makers would need to know the state of the product including aspects such as the conformance of the product to requirements, the usability of the product, any known risks, the product's compliance to any applicable regulations, Software testing enables making objective assessments regarding the degree of conformance of the system to stated requirements and specifications.

8.4 Quality Assurance

Quality assurance consists of the auditing and reporting functions of management. The

goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confident that the product quality is meeting its goals. This is an “umbrella activity” that is applied throughout the engineering process. Software quality assurance encompasses:-

- Analysis, design, coding and testing methods and tools
- Multi-tiered testing strategy
- Control of software documentation and the change made to it.
- A procedure to ensure compliance with software development standards.
- Measurement and reporting mechanisms.

8.4.1 Quality Factors

An important objective of quality assurance is to track the software quality and assess the impact of methodological and procedural changes on improved software quality. The factors that affect the quality can be categorized into two broad groups:

- Factors that can be directly measured.
- Factors that can be indirectly measured

These factors focus on three important aspects of a software product

- Its operational characteristics
- Its ability to undergo changes
- Its adaptability to a new environment.
- Effectiveness or efficiency in performing its mission □ Duration of its use by its customer.

8.5 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures: Interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

Chapter 9

RESULT AND PERFORMANCE ANALYSIS

9.1 Circuit Diagram

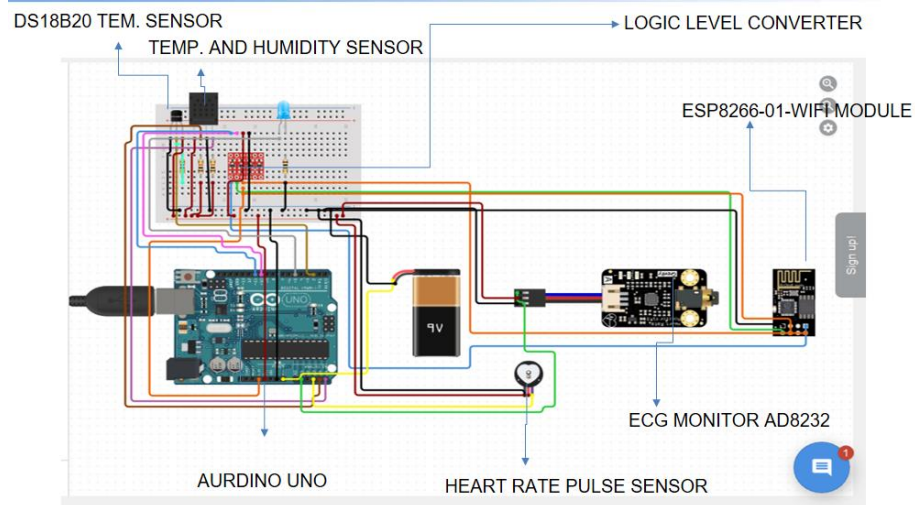


Fig. 9.1 Circuit Diagram

9.2 Database -Firebase cloud storage

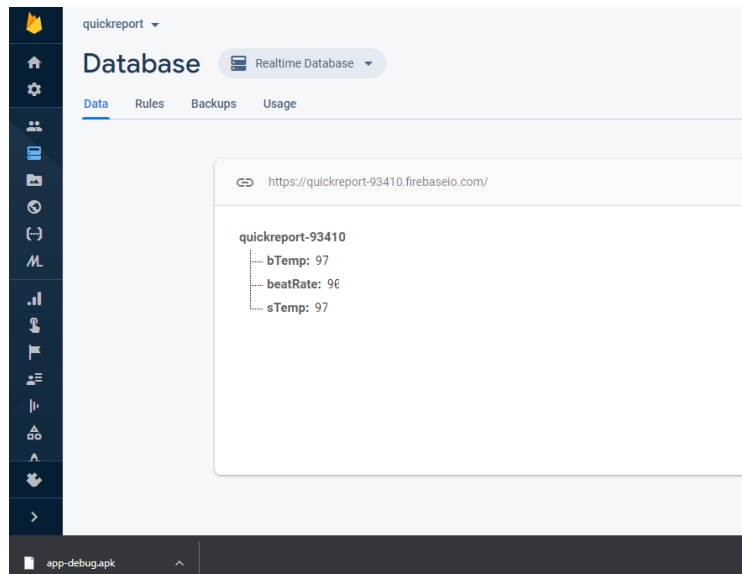


Fig. 9.2 Cloud Storage

9.3 Database -Firebase Authentication rule

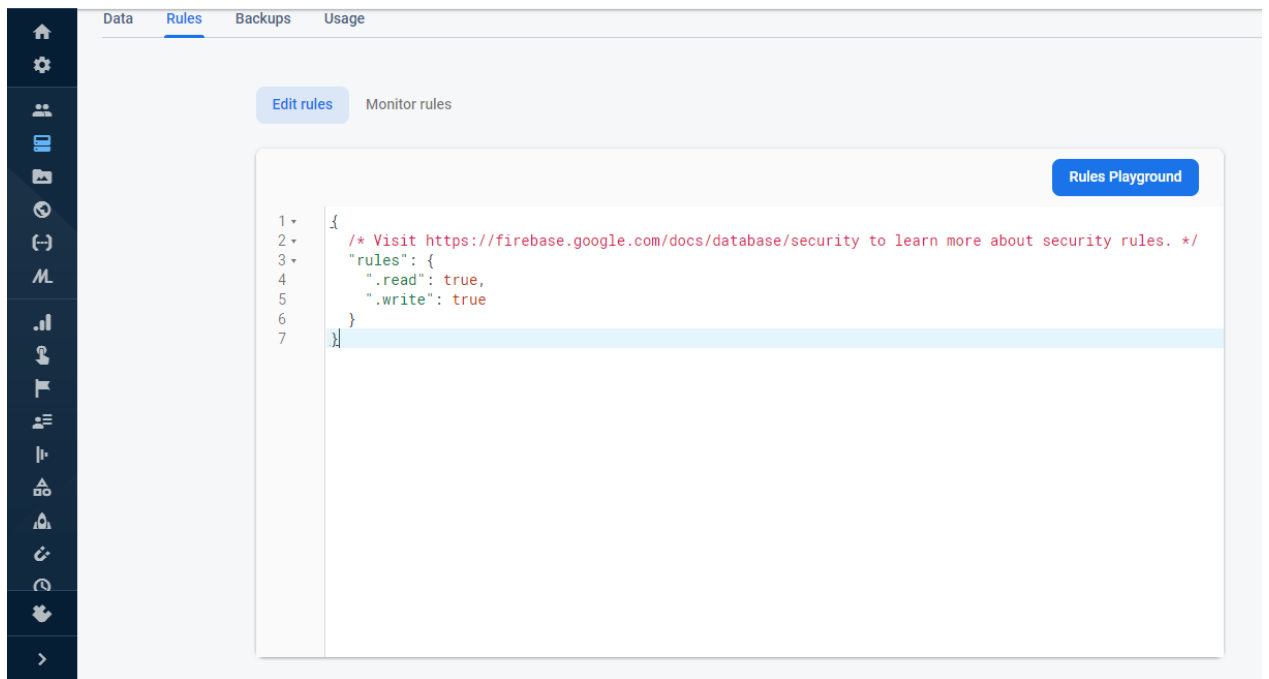


Fig. 9.3 Authentication rule

9.4 Snapshot of Android Application



Fig. 9.4.a Android Splash Screen

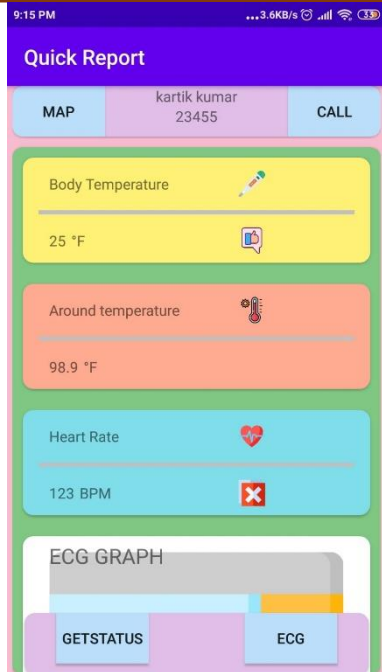


Fig. 9.4.b Android Main Screen1

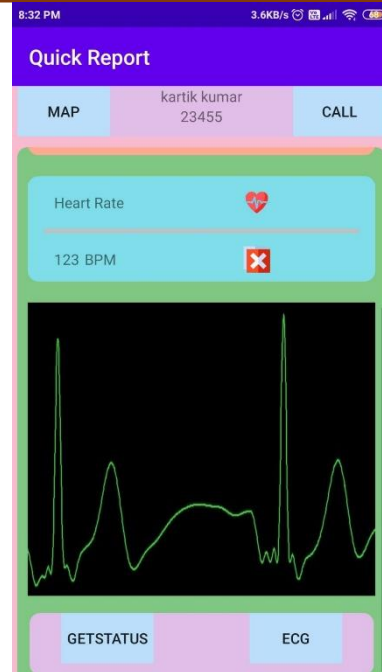


Fig. 9.4.c Android Main Screen2

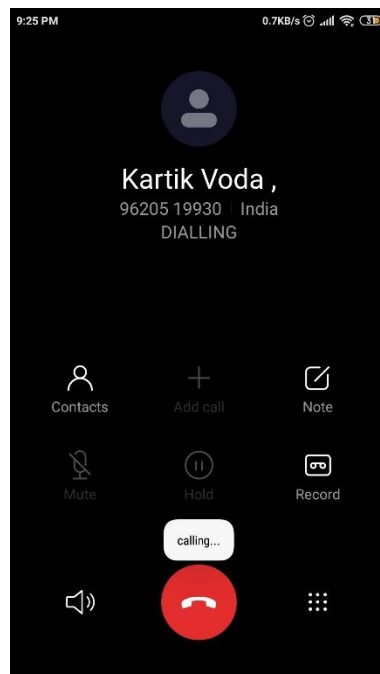


Fig. 9.4.d Android Call Screen

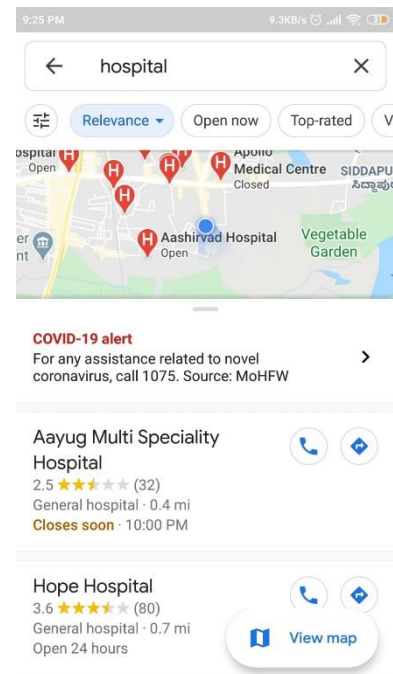


Fig. 9.4.e Android Map Screen

9.5 Snapshot Web Application



Fig 9.5a Web Screen1

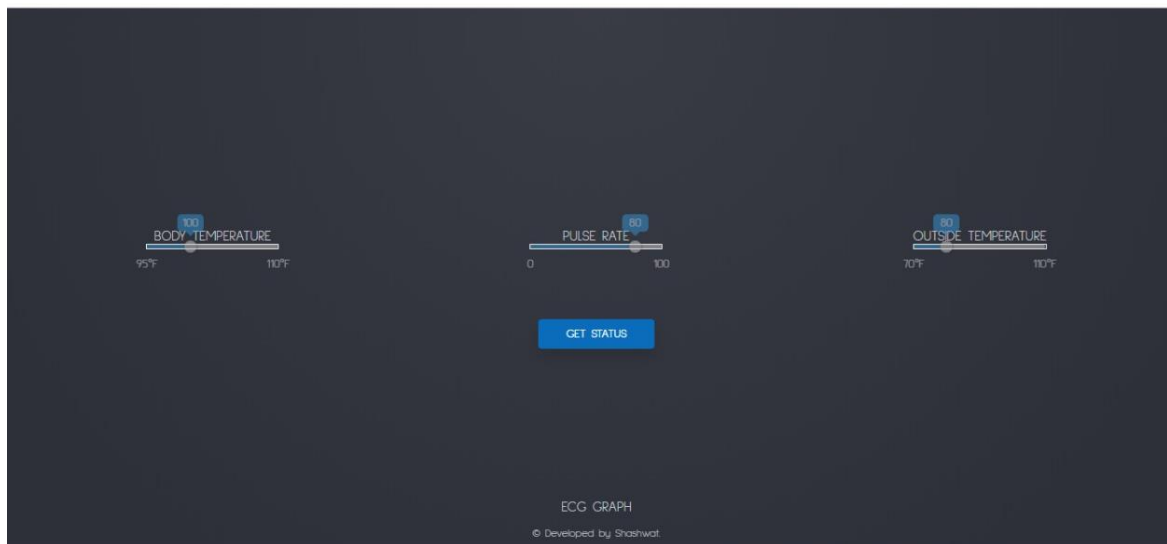


Fig 9.5b Web Main Screen1

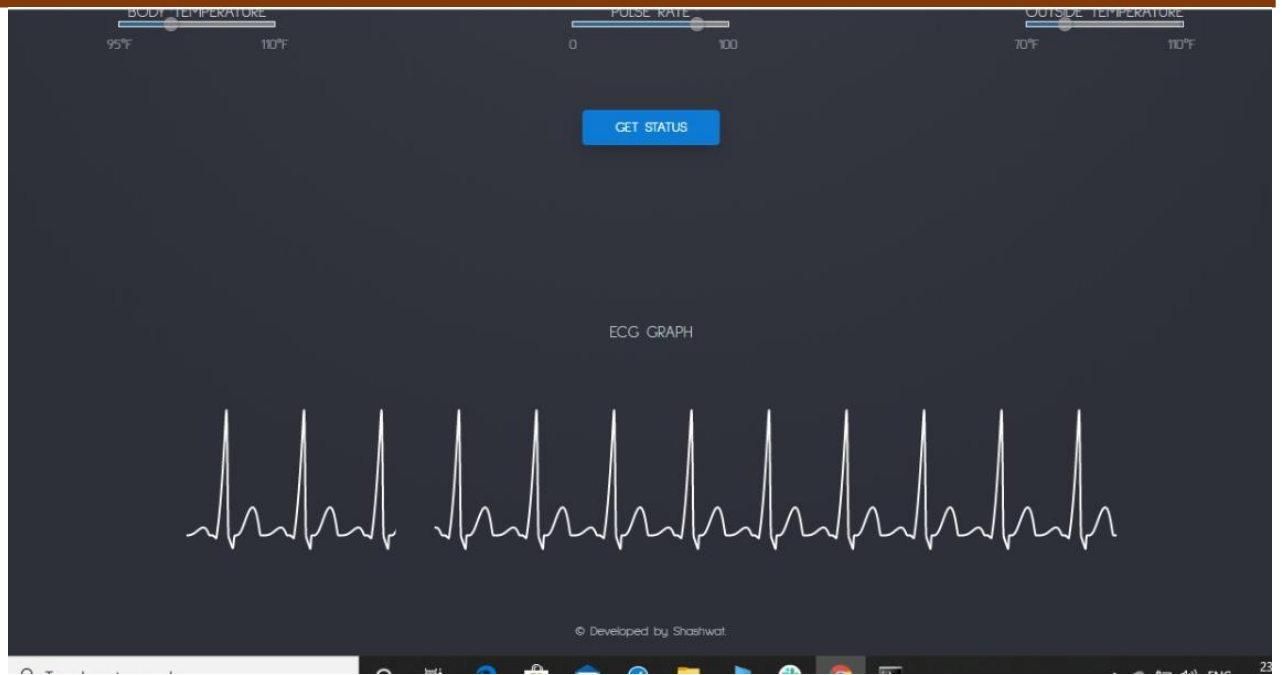


Fig. 9.6c Web ECG Screen

9.6 Snapshot ECG graph

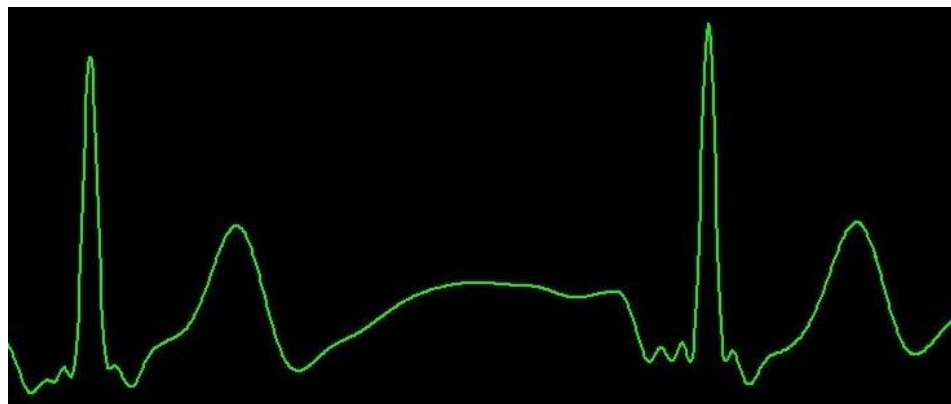


Fig. 9.6 ECG Graph

Summary

This chapter gives a graphic view of the execution of the system. The output screens show each of the execution stages for each Module. Based on the analysis the proposed protocol performs well in terms of accuracy, dependency and usability. The backend and frontend interaction performed to get the required data.

Chapter 10

CONCLUSION AND FUTURE SCOPE

10.1 FUTURE SCOPE

The future development of Intelligent Biomedical Clothing, based on full integration of sensors actuators, energy sources, processing and communication within the clothes, could overcome existing barriers in the use of wearable health systems and be a key enabler technology for cost-effective disease management, disease prevention and overall life style management.

10.2 CONCLUSION

Smart wearables have the potential to offer a minimally obtrusive telemedicine platform for individualised health services that are easily and timely accessible, of better quality for the patient and (citizen than the today solutions, and hopefully cost-effective. The technological development is continuously improving and new challenges are ahead in terms of system and application innovation. In addition, several issues still remain open. Especially, the assessment and evidence proof of added value in respect with cost, access and quality of health services remain challenge due to the difficulty to assess telemedicine applications. In any case, a complete clinical validation off each health wearable application is required.

REFERENCES

- [1] “Smart Insole: A Wearable Sensor Device for Unobtrusive Gait Monitoring in Daily Life”: IEEE Transactions on Industrial Informatics (Volume: 12, Issue: 6 , Dec. 2016).
- [2] “Smart clothes and associated wearable devices for biomedical ambulatory monitoring”: The 13th International Conference on Solid-State Sensors, Actuators and Microsystems, 2005. Digest of Technical Papers. TRANSDUCERS '05.
- [3] “Smart wearable systems for personalized health management: current R&D and future challenges”: Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No.03CH37439).
- [4] “Remote Heart Rate Monitoring System Using IoT”: International Research journal of Engineering and Technology, (Volume: 05, Issue: 04, Apr-2018).
- [5] “The impact of the hybrid platform of internet of things and cloud computing on healthcare systems: opportunities, challenges, and open problems”: Journal of Ambient Intelligence and Humanized Computing, (Dec-2017).
- [6] “Remote Health Monitoring System”, United State Patent, (May-2019).
- [7] “Wireless Body Area Network: A survey on Reliability, Fault Tolerance, and Technologies Coexistence”: ACM Computing Surveys, (Volume: 50, Article: 03, March-2017).
- [8] www.researchgate.net/
- [9] <http://www.arduino.cc>
- [10] <https://www.wikipedia.org/>
- [11] <https://developer.android.com/>
- [12] [https://firebase.google.com/docs/reference/android/com/google/firebase/database/Firebase Database](https://firebase.google.com/docs/reference/android/com/google/firebase/database/FirebaseDatabase)
- [13] <https://www.w3schools.com/html/>
- [14] <https://medium.com/@onyekweretrust/how-to-create-a-simple-graph-in-android-6c484324a4c1> [15]

[16] <https://www.w3.org/Style/CSS/Overview.en.html>

[17] <https://vuejs.org/>

[18] <https://www.circuito.io/app/>